TFC: Gestión de proyectos ágiles

PEC 4 Memoria del proyecto

Autor: Rodrigo Gracia Peña

Consultor: Ana Cristina Domingo Troncho

A mi mujer Raquel, por todo el apoyo recibido durante tantos años y a mi hijo, Alejandro, nacido durante el proyecto.

Marzo 2013

ÍNDICE

1. Metodologías agiles

1.1 Introducción

En este apartado se introducirá en las metodologías agiles, el objetivo principal es realizar un estudio en profundidad sobre las diferentes metodologías agiles más comunes. Además, se trataran puntos más propios y genéricos en una gestión de proyecto de software: Problemáticas, facturación, creación de un proyecto ágil robusto y compañías que han apostado por dichas metodologías.

En este sentido, se tratará las principales metodologías reconocidas por la alianza ágil, y lo que pueden aportar en un proyecto de desarrollo de software. La principal apuesta del agilísimo es realizar procesos iterativos hasta que el cliente disponga del software que necesita. Se apuesta por entregas rápidas, funcionalmente completas, lo que promueve la participación directa del propietario del producto y evitando crear un software no deseado o deficiente.

En conclusión, las metodologías agiles intentan distanciarse de otras más pesadas y que ocasionan lentitud en el desarrollo de proyectos. Por lo tanto, la finalidad del agilísimo es de una forma natural la gestión de personas, para obtener un resultado óptimo, eficaz y rápido de un proyecto.

1.2 Manifiesto ágil

Este manifiesto surgió en 2001 en una reunión celebrada en SnowBird, Utah. Su principal impulsor fue Kent Beck. Aunque muchas de las metodologías que se tratan en este proyecto son anteriores al manifiesto ágil, se puede decir que a partir de la reunión surgió un movimiento que no ha parado de captar adeptos tanto individualmente como empresas. Además, se creó la ñalianza ágilò cuya finalidad es promover el desarrollo ágil en los proyectos.

Manifiesto Ágil

Seguimos estos principios:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

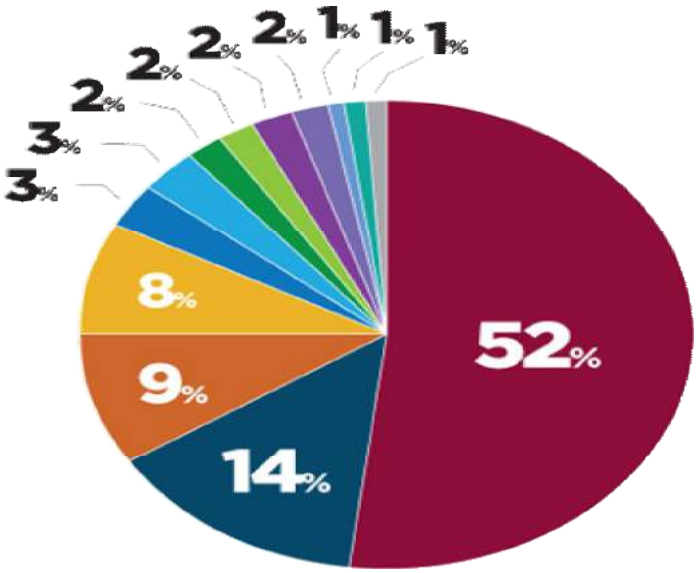
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

1.3 Introducción a las principales metodologías ágiles

1.3.1 Scrum

Sin duda dentro de las metodologías agiles es la más utilizada, por este motivo se realizara una pequeña introducción en este tema, en la segunda parte se detallara en profundidad las características y gestión de proyectos con Scrum.

Versione cada año realiza una encuesta para ver el uso de metodologías agiles en los proyectos de software, la última que disponemos es del 2011. Lo que si nos puede analizar fácilmente que metodologías son las más usadas. En este caso, Scrum se lleva el 52 %.



Scrum 52 %
Fusión entre Scrum y XP (14%)
No sabe (9%)
Hibrido adaptado (8%)

Desde esta grafica podemos sacar unas observaciones interesantes, primero que Scrum predomina sobre el resto de metodologías y segundo la gran flexibilidad y adaptabilidad que proporcionan las metodologías agiles. De este modo, Scrum + XP seria un 14% del porcentaje y híbridos adaptados a proyectos un 8%. En contra, metodologías como XP o Kanban por sí mismas, no consiguen más de un 2 o 3 %.

1.3.1.1 Características

- Se trabaja en iteraciones de 1 a 4 semanas, donde se debe acabar con un producto entregable.
- El equipo es auto organizado, los coordinadores y clientes deben trabajar en todo momento con el equipo de desarrollo, facilitando las tareas y resolviendo dudas.
- Se deben tener unos requisitos perfectamente priorizados reflejando el valor del negocio.
- Se debe mantener un ritmo de trabajo constante, permite que no haya descuidos y retrasos en el sprint.

1.3.1.2 Roles

- Product owner: Dueño del producto
- Scrum master: Es el coordinador del equipo, controla al equipo para la consecución de los objetivos.
- Equipo: Desarrolladores del producto.

1.3.1.3 Reuniones

- Planificaciones del sprint (1 hora por semana/iteración). Debe finalizar con un objetivo claro de lo que se va abordar y con el backlog adecuado, el equipo eligira los ítems que consideran que pueden realizar y los dividirán entre todos.
- Reunión diaria (10-15 min). Cada componente del equipo comenta que está realizando, las tareas que ha terminado y si se tiene alguna dificultad.
- Revisión del sprint (1 hora por semana/iteración). Una vez finalizado el sprint se debe realizar una reunión para comprobar el producto entregado junto con el cliente. Es el momento de saber que se está construyendo.
- Retrospectiva (1 hora). En esta reunión se deben ver que se puede mejorar dentro del equipo, aquí se analiza la forma en la que se está trabajando.

1.3.2 Kanban

Es una palabra Japonesa que literalmente significa ñTarjetas visualesò Estos métodos ya tienen una larga trayectoria en las cadenas de producción, transmitido a desarrollo de software es bastante reciente de 2004.

Es un método visual muy recomendado para gestionar proyectos donde los requisitos cambian constantemente. También es útil en planificaciones y estimaciones de un equipo se alarguen o dejen de ser productivas así como cuando un equipo no se puede comprometer a trabajar en iteraciones fijas.

Se debe disponer del panel de Kanban, donde figuren las etapas de la iteración desde el principio hasta el final.

La primera columna estará el backlog o listado de tareas a realizar, una buena práctica es dividir las tareas en cargas de trabajo similares.

Es importante limitar al número de tareas permitidas por cada columna, de esta forma se eliminan posibles cuellos de botella y los problemas que puedan ocasionar un ritmo constante de trabajo.

Medir el tiempo empleado en una iteración completa, es importante conocer cuándo se ha tardado en realizar la tarea y tomarse su tiempo en analizar cómo reducir ese tiempo. Medir el tiempo también proporciona mayor facilidad para estimar y predecir futuras tareas.

1.3.2.1 Regla 1: Visualizar los estados

Se debe visualizar todo los estados por los que puede pasar una tarea, para ello se utiliza el tablero Kanban donde cada columna será un estado, en cada columna figurara la tarjeta de la tarea correspondiente, de esta forma se sabe de una forma visual y rápida el estado del proyecto y de cada tarea.

1.3.2.2 Regla 2: Limitar el trabajo en progreso

Una de las cualidades que tiene Kanban es el WIP (Work In Progress), esta característica determina cuanto trabajo debe haber como máximo en proceso. Esto permite que no se realicen más tareas, en caso de que por algún motivo alguna encuentre dificultades, puede dedicarse más recursos a resolverlas en vez de aumentar el número de tareas y puedan provocar un cuello de botella.

Medir el WIP:

Si se tiene un WIP muy bajo, podría ocasionar que haya miembros del equipo que no puedan realizar tareas.

En cambio si el WIP es muy alto se corre el riesgo de tener multitareas y de finalizar pocas.

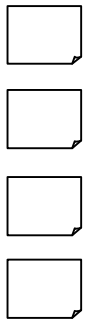
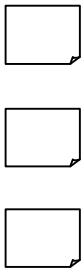
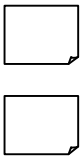
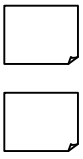
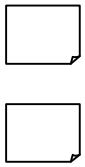
Hay una regla por la que se puede empezar que sería WIP ideal = 2n -1, donde n es el número de miembros del equipo.

1.3.2.3 Regla 3: Medir los flujos de trabajo

Es importante conocer el tiempo que las tareas requieren para finalizarse, es decir han completado todas las fases del tablero. Una vez conocidos los tiempos se va obteniendo el lead time, o tiempo medio en completarse las tareas, el equipo deberá poner esfuerzos en conseguir bajar estos tiempos y ver que dificultades pueden ocurrir en cada tarea.

1.3.2.4 Tablero Kanban

Como se puede observar el número que figura entre paréntesis, es el número máximo de tareas para cada fase del proyecto.

Backlog (4)	Análisis (3)		Construcción (3)		Revisión (4)		Terminado
	En curso	Hecho	En curso	Hecho	En curso	Hecho	
							

1.3.2.5 Teoría de las restricciones en Kanban

La teoría de las restricciones se basa en que la planta de fabricación será tan rápida como el proceso más lento de la cadena de fabricación. Por lo tanto, la teoría de las restricciones tiene como finalidad la búsqueda y eliminación de cuellos de botella.

Se basa en los siguientes puntos:

1- Identificar los cuellos de botella

2- Decidir cómo resolver esos cuellos de botella

3- Subordinar la organización y al sistema para ayudar a resolver la decisión tomada

4- Solventar el cuello de botella

5- Si en los pasos anteriores se consigue resolver el cuello de botella, regresar al paso 1.

1.3.3 Lean software

El siguiente sistema deriva del Sistema de Producción de Toyota, el objetivo principal del sistema de producción y de la metodología de software es eliminar todos los desperdicios (waste). Es decir, todo lo que retrase y no sea necesario para la producción.

Realmente no se podría considerar una metodología por sí misma, son más una serie de principios que se pueden aplicar dentro de los proyectos agiles, son los siguientes:

- Ver todo el conjunto

Se hace hincapié en la importancia de que todos los involucrados en el proyecto, por muy grande que sea, tengan un conocimiento global del mismo. Los proyectos pueden estar formados por varios equipos y diferentes sistemas pero en realidad es un solo proyecto. En este sentido, se debe disponer de una visión integral del proyecto para poder comprobar que puede fallar y donde se debería corregir los problemas. Teniendo una visión holística se dispondrá de un mayor control del proyecto y de sus etapas. Además, si se puede ver el todo es más fácil poder corregir errores en etapas tempranas.

- Eliminar los desperdicios

Todo lo que no añade valor al cliente se considera un desperdicio, en este punto se incluye todas las funcionalidades no necesarias, requisitos poco claros, burocraciaé

El objetivo es encontrar todo aquello que no es necesario para el proyecto y solo supone más carga de trabajo. Para ello se pone el esfuerzo en distinguir y reconocer los desperdicios del proyecto, para este trabajo se utiliza la técnica

ñvalue stream mappingò Una vez localizados los desperdicios, el siguiente paso es eliminarlos. Este proceso se debe repetir iterativamente hasta que lo que parecía en principio esencial pueda ser eliminado.

- Constancia en el aprendizaje

Se fomenta la interacción entre los equipos de desarrollo de trabajo con el cliente. La mejor manera para un enriquecimiento mutuo es el trabajo colaborativo, mediante unas reuniones cortas y constantes para comprobar el estado del proyecto y buscar soluciones conjuntas a los problemas. Este trabajo continuado y colaborativo permite, tanto a clientes como desarrolladores, aprender las necesidades del proyecto y poder atenderlas de una manera más eficaz y sin errores de comunicación. El cliente conoce lo que necesita y el equipo entiende lo que se necesita y cómo se debe afrontar de la mejor manera.

- Decidir lo más tarde posible

La intención es retrasar las decisiones cuando realmente están claras, es decir no es necesario realizar una previsión temprana de lo que hay que realizar. Debido a los procesos iterativos de las metodologías agiles se pueden resolver los problemas cuando realmente son necesarios, sabiendo a ciencia cierta lo que se requiere por parte del cliente y como hay que afrontarlo.

- Reaccionar tan rápido como sea posible

Requiere de unas entregas de funcionalidades acabadas y sin fallos en iteraciones cortas, esto provoca un mejor aprendizaje y comunicación dentro del equipo. Con unas entregas de software rápidas y de calidad el cliente asegura sus necesidades actuales y les permite obtener tiempo para planear necesidades futuras. Además, de disponer de una confianza y seguridad en el equipo de desarrollo, se buscan resultados lo antes posible.

- Potenciar el equipo

Es importante que los integrantes de los equipos tengan iniciativas y no sean simplemente recursos con tareas a cumplir. Se debe fomentar la comunicación directa entre los desarrolladores y directivos e incluso puedan proponer sugerencias. Por lo tanto se incentiva la motivación y la participación de los miembros del equipo, que colabore y tenga acceso a los clientes, en esencial que

el desarrollador sea una pieza importante dentro de los proyectos y sea una parte esencial y activa dentro de los proyectos.

- Crear la integridad

Percepción de integridad: Las cualidades generales que el cliente debe conocer sobre el proyecto: como se implementa, como es de accesible e intuitivo, cuál es su coste, que debe contener...

Integridad conceptual: Significa que las diferentes partes por separado funcionan perfectamente una vez unidos, realizando un proyecto único con una fuerte integración. Facilitando el desarrollo, el mantenimiento y la eficiencia.

Se debe prestar atención hacia una arquitectura integrante, donde la refactorización debe aportar sencillez, claridad y cantidad mínima de funcionalidades. Además, se enfatiza el proceso de construcción debe estar automatizado incluido las pruebas. Al final, la integración debe ser garantizada con unas pruebas globales que garanticen lo que se espera del producto. Las pruebas deben de estar bien diseñadas y si no generan valor deben desecharse. En este sentido, el testing forma parte de la producción del software

1.3.4 Xtreme programming

Su impulsor fue Kent Beck, el cual fue uno de los firmantes y de los principales impulsores del manifiesto ágil.

1.3.4.1 Historias de usuario

Las Historias hacen la función de casos de uso y de requisitos pero no son lo mismo, es una breve descripción de lo que el cliente quiere más o menos tres frases sin contenido técnico. Esto es así, porque realmente cuando llegue el momento de afrontarlas se debe estudiar con el cliente directamente en qué consisten y que es lo que hay que hacer, realizando un estudio más detallado.

Las Historias de Usuario deben ir acompañadas siempre por unas pruebas de aceptación, para verificar una vez terminadas que son correctas. Si las Historias

superan la estimación de tres semanas máximo de desarrollo se debe descomponer en Historias más sencillas.

A continuación se describen unas buenas prácticas a seguir (Kuphal 2011):

1- Las historias suelen ser una serie de tareas, donde cada tarea no debe superar más de tres días de desarrollo. En caso contrario, se deben dividir en tareas más pequeñas.

2- Crear tareas que una vez completadas generen un producto entregable, es decir tareas relacionadas con una misma funcionalidad.

3- No dedicar excesivo tiempo a estudiar todos los detalles de cada tarea. Es verdad que puede ayudar a realizar una estimación más precisa pero retrasa el proceso de división de las Historias de Usuario en tareas.

4- En el conjunto de tareas deben estar incluidas las tareas de pruebas y su posible automatización.

1.3.4.2 Roles XP

- Programador. El programador escribe las pruebas unitarias y produce el código del sistema.

- Cliente. Escribe las historias de usuario y realiza las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

- Encargado de pruebas (Tester). Es el encargado de las pruebas del proyecto, ayuda al cliente a realizar las pruebas funcionales.

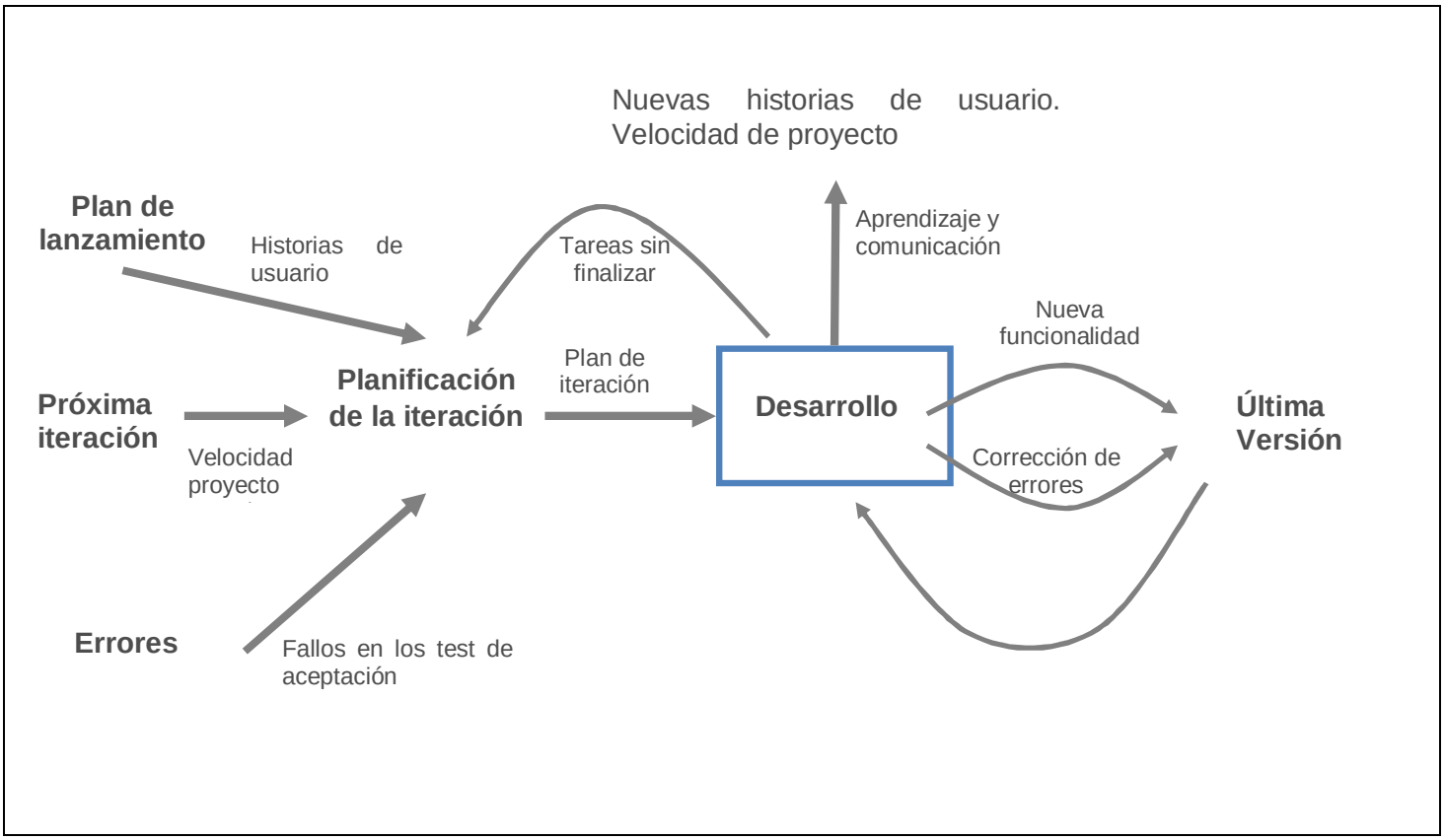
- Encargado de seguimiento (Tracker). Es el encargado de verificar que las tareas que se realizan van acorde con lo estimado, lleva el control de tiempos diario. Debe adelantarse a posibles desviaciones imprevistas.

- Entrenador (Coach): Es el máximo responsable del proyecto, es el encargado de garantizar que el cumplimiento de los hitos se lleva a cabo.

- Consultor: Es un especialista, externo al equipo se puede requerir para solventar cualquier imprevisto relacionado con su especialidad.

- Gestor (Big boss): Es el coordinador, el enlace entre el equipo de desarrollo y los clientes, controla de que el equipo trabaje eficazmente creando las condiciones adecuadas.

1.3.4.3 Iteración en XP



Al principio de cada iteración se debe establecer, por parte del cliente, las tareas más prioritarias según las Historias de Usuario disponibles y los errores producidos en los test de aceptación, todas estas tareas irán al plan de lanzamiento.

Una vez establecidos las prioridades se deben estimar por el desarrollador cuantos días de programación requieren 1, 2 y 3. Se puede añadir ½ si se requiere, las tareas de menos se pueden agrupar en una sola y las tareas de más de tres días se deben dividir en tareas más pequeñas. Una vez estimadas, se dispone del plan de iteración con las tareas a incluir, hay que tener en cuenta que las iteraciones en XP se aconsejan que sean de 1 a 3 semanas.

Al establecer el plan, comienza la fase de desarrollo hasta la fecha fin del mismo, puede que al finalizar no se cumplan con todas las tareas estimadas, las cuales pasan a la siguiente iteración. Dentro de la fase de desarrollo pueden surgir nuevas Historias de Usuario o al pasar los test de aceptación haya fallos, están nuevas tareas no contempladas anteriormente deben ser informadas al cliente para que les dé una prioridad.

1.4 Grandes compañías que usan metodologías ágiles

El incremento de las metodologías agiles, también es debido a su uso por grandes compañías tecnológicas, han visto como las metodologías tradicionales eran demasiado pesadas y rígidas para el desarrollo de su software. Por este motivo, para estas empresas se requiere agilidad en sacar un producto por dos motivos principales:

1. Si se demora en el tiempo, es posible que la competencia se haya adelantado y el producto quede obsoleto

2. Es más fácil sacar betas y aplicaciones con una mínima funcionalidad para comprobar el interés de los usuarios, en caso de poca aceptación es más fácil cerrar el proyecto con un gasto mínimo antes que terminarlo por completo.

Google y Yahoo han apostado por las metodologías agiles y realmente consiguen unos éxitos importantes. A continuación se presenta un cuadro con compañías que usan Scrum.

Sectores	Ejemplos de empresas que utilizan metodologías ágiles como Scrum
Media y Telcos	BBC, BellSouth, British Telecom, DoubleYou, Motorola, Nokia, Palm, Qualcomm,Schibsted, Sony/Ericsson, Telefonica I+D, TeleAtlas, Verizon
Software, Hardware	Adobe, Autentia, Biko2, Central Desktop, Citrix, Gailén, IBM, Intel, Microfocus, Microsoft, Novell, OpenView Labs, Plain Concepts, Primavera, Proyectalis,Softhouse, Valtech, VersionOne.
Internet	Amazon, Google, mySpace, Yahoo

ERP	SAP
Banca e Inversión	Bank of America, Barclays Global Investors, Key Bank, Merrill Lynch
Sanidad y Salud	Patientkeeper, Philips Medical
Defensa y Aeroespacial	Boeing, General Dynamics, Lockheed Martin
Juegos	Blizzard, High Moon Studios, Crytek, Ubisoft, Electronic Arts
Otros	3M, Bose, GE, UOC, Ferrari

1.5 Fusión de metodologías y modelos.

Este apartado se ha considerado de interés debido a la gran cantidad de combinaciones posibles que se pueden realizar con las metodologías agiles. La gran flexibilidad y las pocas normas que contienen permiten que varias metodologías puedan convivir en un mismo proyecto o que un proyecto forme su propia metodología de trabajo.

Personas más estrictas con las normas pueden no estar de acuerdo con estas fusiones, pero si se es consciente de los objetivos que se persiguen, metodologías de trabajo rápidas y poco restrictivas, podrían ser perfectamente válidas.

En este sentido, el Scrum-ban puede ser utilizado para ir adentrando los equipos al sistema de metodología agiles y ciclos iterativos, a la vez se van añadiendo particularidades propias de Scrum como las reuniones diarias de 15 min. De este modo, tanto desarrolladores como stakeholders se van acostumbrando de una manera más gradual al nuevo método de trabajo.

Como se observa la flexibilidad de las metodologías puede ocasionar infinidad de variantes, ahora bien hay una serie de normas que siempre se deben cumplir: Ciclos cortos y fijos de entregas, donde el producto este acabado y probado, sin olvidar la calidad en el software.

1.6 Posibles soluciones de facturación para un proyecto ágil

A simple vista que complicación debería tener un proyecto ágil a la hora de facturar, la respuesta es muchas. Nos encontramos con problemas de base como se puede facturar un proyecto que no tiene fecha de finalización establecida y unos requisitos cerrados.

Normalmente cuando sale adelante un proyecto se abre un pliego con una cantidad de dinero fija, unos requisitos y unos plazos que cumplir, así que este método no funcionaría correctamente con las características de un proyecto ágil real, no hay proyectos ni requisitos cerrados (flexibilidad), y si se realiza de este modo se perdería el valor de enriquecimiento que proporciona estas metodologías, por lo cual vamos a estudiar posibles soluciones a estos problemas.

1. Facturaciones fijas por tiempo. El proveedor estima que cuesta el equipo de desarrollo por meses, el cliente paga por los meses que considera oportuno o las entregas requeridas hasta tener el producto adecuado.

2. Un equipo de X personas elegidos entre el cliente y la empresa proveedora, cada uno de ellos con un coste establecido (body shopping). Similar al primer caso, se dispone de X recursos por la cantidad de meses que se requieran hasta tener un proyecto adecuado a las necesidades del cliente. El body shopping se suele dar en empresas que subcontratan desarrolladores, no suelen ser personal de la empresa proveedora.

3. Llave en mano, otro tipo de contrato clásico y que realmente no cuadran para metodologías agiles, en estos casos normalmente quien sale perjudicado es el proveedor al tener que estimar todas las Historias de Usuario iniciales que surjan, pero que ocurre si estas Historias cambian. La solución que se da en estos casos es estimar las Historias según un peso o un valor, si el usuario quiere cambiar una vez iniciado el proyecto se puede realizar, pero debe sustituirse por otra de similar peso en las iniciales.

En realidad, puede haber muchos más formatos, muchos de ellos fusión entre contratos cerrados y por tiempos, con penalizaciones y bonificaciones. Pero lo principal de las metodologías agiles radica en la confianza entre proveedores y clientes, y por lo tanto el contrato debería ser de colaboración y de beneficio mutuo.

1.7 Dificultades y factores críticos de éxito en la implantación de metodologías agiles

Las dificultades que suele encontrar para implantarse Scrum en los proyectos, suelen ser múltiples, pero principalmente Factor humano y económico.

Dentro del factor humano, se encuentra la resistencia al cambio, las personas que llevan realizando sus tareas de una forma durante muchos años, normalmente no suelen tener predisposición a adoptar nuevas métodos de trabajo. Esto aplica tanto a los desarrolladores como a los stakeholders, por este motivo hay que concienciar realmente en que consisten estas metodologías y el potencial que disponen.

En el factor económico, como se ha definido en el punto anterior los proyectos agiles por norma general no predefinen una fecha fin, así como unos requisitos cerrados. En este sentido, el principal problema que se encuentran es que los clientes suelen disponer de un gasto prefijado el cual no deberían salirse sin una buena justificación, por lo tanto es complicado negociar un contrato donde realmente los requisitos no están fijados y la fecha fin del proyecto no está establecida.

Estos dos puntos anteriores, aun siendo solo dos son suficientes para que las metodologías agiles, por lo menos en España, no se hayan arraigado en el desarrollo de software.

Para los factores críticos de éxito se organizan en 5 categorías: Organización, personal, proceso, tecnología y proyecto (Chow & Chao 2008).

Dimensión	Factor crítico de éxito
Organización	Apoyo a la directiva
	Ayuda al sponsor o manager
	Cultura cooperativa
	Alto valor al ñcara a caraò
Personal	Equipos competentes y experimentales
	Equipos motivados

	Jefes de equipo con altos conocimientos en metodologías agiles
	Buena relación con el cliente
Proceso	Seguimiento de gestión ágil de requisitos
	Seguimiento de gestión ágil de proyectos
	Seguimiento de gestión ágil de configuración
	Fuerte comunicación focalizada en el cara a cara
	Presencia regular del cliente
Tecnología	Establecimiento de estándares de programación
	Actividades rigurosas de refactorización
	Preparación técnica de equipo
	Correcta documentación
Proyecto	No critico
	Dinámico
	Equipos pequeños
	Requisitos cambiantes

2. Scrum al detalle

2.1 Introducción

En este apartado, se definirán todas las características propias de la metodología ágil más común y extendida, Scrum. El principal objetivo es explicar de una manera clara los distintos mecanismos y reglas que deben aplicarse para poner en práctica un proyecto Scrum.

El principal hito de Scrum es gestionar proyectos basados en pequeños grupos de trabajo, de cuatro a nueve personas, desarrollándolo de una forma iterativa y constante. Para un buen funcionamiento se necesita de una comunicación fluida y diaria, de una mejora continua y la apuesta por la calidad del código de todos los miembros del equipo. Son normas muy sencillas, muchas de ellas de sentido común, pero no hay que olvidar que precisamente la apuesta por lo sencillo y lo ágil es lo que proporciona mayor valor a Scrum.

En conclusión, Unas normas sencillas de gestión hace posible una organización rápida y sin demasiados obstáculos, es cierto que las pocas normas que dictamina Scrum se deben cumplir sin excepciones (Iteraciones cortas y constantes, reuniones diarias la misma hora y mismo lugar, no excederse en los tiempos de reunionesé) en cualquier otro caso, no se esta siguiendo la metodología.

A continuación, se aporta una descripción formal de Scrum según Ken Schwaber and Jeff Sutherland, en The Scrum Guide (www.scrum.org).

Scrum se fundamenta en la teoría empírica de control de procesos, o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Scrum emplea una aproximación iterativa e incremental para optimizar la predictibilidad y controlar el riesgo.

Tres pilares soportan toda implementación del control empírico de procesos: transparencia, inspección y adaptación.

Transparencia

Los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. La transparencia requiere que dichos aspectos sean definidos por un estándar común, de modo que los observadores compartan un entendimiento común de lo que se está viendo.

Por ejemplo:

Todos los participantes deben compartir un lenguaje común para referirse al proceso; y,

Aquellos que desempeñan el trabajo y aquellos que aceptan el producto de dicho trabajo deben compartir una definición común de ñechoò

Inspección

Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso hacia un objetivo, para detectar variaciones no deseables. Su inspección no debe ser tan frecuente como para que interfiera en el trabajo. Las inspecciones son más beneficiosas cuando son realizadas de forma diligente por inspectores expertos, en el mismo lugar de trabajo.

Adaptación

Si un inspector determina que uno o más aspectos de un proceso se desvían de límites aceptables, y que el producto resultante no será aceptable, el proceso o el material que está siendo procesado deben ser ajustados. Dicho ajuste debe ser realizado cuanto antes para minimizar desviaciones mayores.

Scrum prescribe formalmente cuatro oportunidades para la inspección y adaptación, tal y como se describen en la sección Eventos de Scrum del presente documento.

- *Reunión de Planificación del Sprint (Sprint Planning Meeting)*
- *Scrum Diario (Daily Scrum)*
- *Revisión del Sprint (Sprint Review)*
- *Retrospectiva del Sprint (Sprint Retrospective)*

2.2 Roles

2.2.1 Product Owner (Propietario del Producto)

Características	Es el máximo responsable de las funcionalidades que se deben incluir en el proyecto y de darles una prioridad para generar el product backlog. Decidirá qué se debe incluir en cada Sprint, Será el encargado de gestionar todas las ideas y requisitos de las diferentes áreas implicadas que tengan algún interés en el proyecto.
Funciones y responsabilidades	Expresar y clarificar las funcionalidades del product backlog Decisión final sobre las funcionalidades a implementar Dar una prioridad a las funcionalidades y mantener el product backlog. Implicación en el proyecto. Conocimiento del negocio y del proyecto. Revisión del producto para que se adecue a las necesidades reales de la empresa y lo solicitado

2.2.2 Scrum Master

Características	El scrum master es el encargado de gestionar el proyecto, su principal objetivo es verificar que las diferentes etapas del sprint se van cumpliendo y evitar en la medida de lo posible cuellos de botella o dificultades en el desarrollo.
Funciones y responsabilidades	Deberá trasmitir correctamente Scrum tanto al producto owner como al equipo. Controlará que en el proyecto cada uno tenga su papel, y no puedan interferir terceras personas Previsión de problemas y posibles cuellos de botella. Resolución de los problemas surgidos dentro del proyecto. Fomentar un clima de trabajo colaborativo y autogestionado Motivar al equipo

2.2.3 Scrum Team

Características	El scrum team son los desarrolladores del producto, normalmente se compone por personal con diferentes perfiles que abarque las posibles áreas y tecnologías del proyecto. (Programadores, diseñadores, arquitectos, testers).
Funciones y responsabilidades	Llevar el Backlog de producto, a desarrollos potencialmente funcionales y operativos. Asegurar la calidad en los productos

2.2.4 Otros Roles

Hay otros roles que no se consideran parte del proceso Scrum, pero disponen de un papel relevante y se deben tener en cuenta. Una parte de Scrum es poder involucrar en el proyecto todos los interesados que de una manera u otra, tienen algún tipo de vinculación, ya sean usuarios, expertos del negocio (Stakeholders), gerentes... Es importante, que las partes implicadas puedan aportar sus opiniones y generar retroalimentación para poder completar y planear cada sprint. De este modo, el equipo puede comprobar que los avances corresponden con lo que se espera y necesitan.

2.2.4.1 Usuarios

Principalmente son los destinatarios finales del producto, quienes realmente van a utilizarlo. Por este motivo, la implicación de los usuarios en fundamental en la creación del software, se necesita saber con regularidad que el proyecto se ajusta a las necesidades reales del usuario y es lo que se espera.

2.2.4.2 Stakeholders (Clientes, Proveedores)

Son principalmente las personas que tienen un interés económico del proyecto, suelen ser los que hacen posible el proyecto gestionando la parte económica. Generalmente son gerentes o directivos, solo participan en las revisiones de los sprint, de este modo pueden ver los avances de los proyectos.

2.3 Formación de equipos

Para una formación de equipos Scrum, no se debe separar por tecnologías es decir, desarrollo de BBDD, desarrollo de aplicaciones, desarrollo de pruebasé Sino que los equipos deben ser multifuncionales, autosuficientes capaces de desarrollar y probar una característica del producto, sin olvidar la calidad. El equipo deberá incluir diferentes perfiles capaces de tomar una historia del backlog y transformarlo en un producto.

El tamaño de un equipo en Scrum debe encontrarse entre cinco y nueve personas, menos de cinco el equipo está expuesto a cualquier imprevisto que surja por lo tanto no pueda cumplir con los objetivos fijados en el sprint. Más de nueve personas la comunicación y colaboración entre los miembros es más complicada con riesgos de crear subgrupos.

El equipo debe ser autoorganizado, comunicativo y todos deben confiar en el resto de compañeros. Además, Los miembros del equipo deben disponer de las habilidades necesarias para poder afrontar las tareas encomendadas en cada sprint. Evitar que se dependan de terceras personas para que no se vean comprometidas las tareas de cada sprint. Se debe fomentar el trabajo colaborativo, compartiendo habilidades y experiencias, generando un equipo integrado y eficaz.

Los miembros del equipo deben dedicarse al proyecto a tiempo completo para evitar dañar su productividad por cambios de tareas en diferentes proyectos, para evitar interrupciones externas y así poder mantener el compromiso que adquieren en cada sprint. Además, los miembros del equipo deben trabajar en la misma localización física, para poder disponer de una comunicación directa. El equipo debe ser estable durante el proyecto, sus miembros deben cambiar lo mínimo posible, para poder aprovechar el esfuerzo que les ha costado construir sus relaciones interpersonales, engranarse y establecer su organización del trabajo.

Realizaran las siguientes tareas de manera conjunta:

- Decidir que historias del backlog pueden realizar en el sprint.
- Estimar la complejidad de cada historia de backlog.
- En la reunión de planificación del sprint, se decide cómo se va a realizar el trabajo

- Seleccionar los requisitos y realizar al cliente las preguntas necesarias.
- Identificar todas las tareas necesarias para completar cada requisito.
- Estimar el esfuerzo necesario para realizar cada tarea.
- Cada miembro del equipo se autoasigna a las tareas.
- Durante el sprint, trabajar de manera conjunta para conseguir los objetivos de la iteración.
- Cada especialista lidera el trabajo en su área y el resto colaboran si es necesario para poder completar un requisito.

Al finalizar el sprint:

- Mostrar al cliente, usuarios y resto de equipos los requisitos completados.
- Hacer una retrospectiva al final de cada sprint para mejorar de forma continua su manera de trabajar.

2.4 Reuniones

2.4.1 Scrum daily meeting

Es la reunión diaria del proyecto, se realiza el seguimiento de las tareas y se comprueba que no haya dificultades para cumplirlas, tiene unas particularidades que se deben tener en cuenta:

1. Las reuniones no pueden superar los 15 min
2. Las reuniones se celebraran siempre a la misma hora y en el mismo lugar.
3. Los miembros del equipo deberán permanecer de pie, para no alargar la reunión.

En las reuniones, el Scrum Master debe verificar el avance del sprint, para ello realizara las siguientes preguntas a cada uno:

- ¿Qué has hecho desde ayer?
- ¿Qué es lo que estás planeando hacer hoy?
- ¿Has tenido algún problema que te haya impedido alcanzar tu objetivo?

2.4.2 Reunión de Planificación del Sprint (Sprint Planning Meeting)

Es la reunión inicial de cada sprint, en esta encuentro se acuerda que debe entrar dentro del ciclo. En este sentido, se debe identificar el trabajo posible que podrá realizarse y preparar así el Sprint Backlog y se acordara entre todos el tiempo de cada tarea. La reunión no puede durar más de ocho horas

2.4.3 Reunión de Revisión del Sprint (Sprint Review Meeting)

Esta reunión se realiza al finalizar el sprint, se debe hacer una evolución de las tareas completas y las que no se han podido completar, en este caso analizar el problema que ha impedido su cumplimiento.

Se prepara el trabajo completado para mostrarlo al resto de equipos, stakeholdersé que tengan un interés en el proyecto, solo se debe mostrar la funcionalidad totalmente completada. La reunión no debe durar más de cuatro horas.

2.4.4 Retrospectiva del Sprint (Sprint Retrospective)

Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua del proceso, es decir que se puede mejorar en calidad y funcionalidad para los siguientes sprints. Esta reunión tiene un tiempo fijo de cuatro horas.

2.4.5 Scrum de Scrum

Es el método de trabajo para organizar varios equipos de trabajo para desarrollos de grandes proyectos con Scrum, hay que recalcar que en este tipo de proyectos la organización interna y su interacción con el resto debe ser exhaustiva y cualquier malentendido puede provocar retrasos importantes, esto no es intrínseco de scrum sino de cualquier tipo de organización donde varios equipos estén trabajando simultáneamente.

La técnica consiste en disponer de varios equipos y utilizar reuniones Scrum de Scrum para coordinarse, a cada reunión debe asistir uno o dos integrantes de cada equipo.

A continuación, se van a tratar los siguientes apartados: Cuantos equipos y tamaño se necesitan, especialistas que debe tener cada equipo, quienes deben asistir a las reuniones, como se escalan los puntos tratados y la frecuencia de los meeting.

Equipos y tamaño

Como se ha mencionado en el punto de equipos, cada grupo debe tener un mínimo de cinco hasta nueve miembros, cuando se tienen más integrantes se dividen los equipos para conseguir el número 5-9, cada equipo debe tener su Scrum Master, su propio backlog y tener un Product Owner.

Asistentes a las reuniones

Cada equipo designa a un integrante del equipo para asistir a las reuniones intergrupales, se recomienda que sea un perfil técnico y no el Scrum Master o el Product Owner. No necesariamente tiene que ser siempre la misma persona, sino que puede rotar según la etapa del proyecto y quien esté en mejor posición para contribuir con la reunión. Al principio del proyecto deberían enviarse a personal técnico o diseño de usuario, mientras que al final debería enviarse a personal que está ejecutando las pruebas.

Si el número de equipos Scrum es pequeño, pueden asistir hasta dos personas por equipo, por ejemplo un participante técnico y el Scrum Master.

Escalamiento de las reuniones

Cuando los proyectos son grandes, las reuniones Scrum de Scrum se pueden escalar en múltiples niveles. Por ejemplo, supongamos que se tienen 49 equipos Scrum de 7 personas cada uno. Por cada equipo se selecciona a una persona a asistir a reuniones Scrum de Scrum, de esta forma, se conforman 7 reuniones con 7 participantes cada una. Por cada uno de estos equipos se puede escalar a 3 equipos y estos a su vez a un equipo directivo. Bajo este ejemplo se tendrían 4 niveles.

Frecuencia de las reuniones

Las reuniones deben ser frecuentes, diaria o tres veces por semana. Se recomienda que tenga la misma duración de la reunión diaria (15 minutos). Aunque es recomendable disponer de 30 minutos más para resolución de problemas y consultas que surjan, en esta reunión se debe evitar dejar temas pendientes sin resolver, para tener que volver a reunir a las mismas personas.

Agenda

La agenda de las reuniones debería ser similar las reuniones diarias. A cada integrante que representa cada uno a un equipo, se le pregunta lo siguiente:

¿Qué actividades ejecutado tu equipo desde la última reunión?

¿Qué actividades realizará tu equipo antes de la próxima reunión? (Próximos pasos).

¿Existen impedimentos en tu equipo?

¿Existe alguna actividad a ejecutar próximamente por tu equipo que interfiera o afecte de alguna forma el trabajo de otro equipo?

Al igual que las reuniones diarias (Daily Scrum), la intención es que la exposición de cada integrante sea breve. Los problemas se pueden mencionar, no se buscan soluciones hasta que cada integrante ha hecho su exposición.

A continuación, se realiza la segunda parte de la reunión, en la cual los participantes discutirán los problemas o situaciones planteadas, o puntos tratados en reuniones previas que aún no se han cerrado. Además, El equipo mantendrá un Backlog de Scrum de Scrum, un listado con los asuntos que han tenido que ser tratados

2.5 Sprints

Los sprint son las iteraciones cortas que se usan para completar las historias de usuario y generar un producto entregable. Se definen los sprint duren entre 2 y 4 semanas, realmente es el ciclo de trabajo, y el punto esencial del desarrollo de Scrum, donde se debe poner en práctica toda la teoría y seguir los pasos y normas de Scrum. (Comunicación directa, reuniones diarias, calidad de los desarrollos).

Se establece que entre dos y cuatro semanas, es un tiempo adecuado para entregar un producto acabado, más tarde en próximos sprint se podrán añadir más funcionalidades, pero en cada sprint se debe generar un producto entregable. Menos de dos semanas de sprint es demasiado corto para disponer de productos entregables, más de un mes se va perdiendo la anticipación a elementos no deseados o detección de problemas.

Para la consecución de objetivos, se deben cuidar determinados aspectos:

- Estabilidad: No se puede cambiar el contenido o variar el objetivo, puede haber una sola excepción que sea beneficioso y esté justificado, por ejemplo una historia de usuario no sea necesaria o tener que realizar una historia que bloquea a otra.

- Obtención de resultados: El objetivo de cada sprint es realizar un producto parcialmente desarrollado, este producto puede ir creciendo de forma continua a lo largo de los sprints.

- Mejora continua: Se dedica un tiempo importante para identificar y aplicar mejoras en la forma de trabajar, en la productividad y calidad. En la retrospectiva del sprint se deben tratar que puntos de deben mejorar y como se van a afrontar en los siguientes sprints.

- Anticipación: Hay que aprovechar los cambios entre sprint para ver adelantarse e ir tratando de los retos, cambios y problemáticas de la siguiente iteración.

2.5.1 Sprint 0

El sprint cero se dedica principalmente a preparar el equipo, comprobar las primeras impresiones, comprobar el backlog inicial, ver cómo afrontar los primeros desarrollos. Se detallan los puntos que se podrían tratar en un sprint cero:

- Quien se involucra en el proyecto (Stakeholders, miembros del equipo)
- Como se organizara el proyecto
- Cuáles son los principales riesgos y limitaciones
- Cuál es el alcance y propósito del proyecto. Visión
- Que se va a desarrollar, primeras historias de usuario
- Preparar el grafico burn-down y la velocidad objetivo
- Documento de arquitectura a alto nivel, como se va a construir
- Cuánto dinero se necesita
- Preparar los entornos, pruebas, implementación y entornos de integración continua

2.6 Backlogs

2.6.1 Product Backlog

El Product Backlog se podría definir como los requisitos que se necesitan, no es una lista de requisitos cerrada, en cualquier momento el Product Owner puede quitar o añadir requisitos a la lista, eso sí es su obligación que estos requisitos los tenga más o menos claro para poder explicarlos y además lo tenga clasificados y ordenados por prioridad.

Cuando el requisito es funcional hablamos de Historias de Usuario. En cambio sí es operacional, es decir no genera negocio pero es necesario para que el proyecto funcione o mejore en calidad y rendimiento, entonces no se denomina de historias de usuario, sino simplemente requisito operacional.

Par comprobar si los ítems están bien realizados, se puede utilizar la regla de las 3C:

- Card: La redacción debe caber en una cartulina pequeña, esto quiere decir que es una breve descripción y que describe lo suficiente para poder entenderla.

- Conversation: Debe ser el resultado entre conversación y negociación entre el responsable del producto y el equipo. Es decir, ha habido un acuerdo y un dialogo previo para entender en que consiste el requisito.

- Confirmation: Los elementos deben ser de fácil conformación, deben contener un criterio para indicar cuando realmente se han cumplido y finalizado.

Para comprobar que una historia de usuario está bien definida, debe cumplir la regla INVEST:

- Indepedent: No depender de otras historias de usuario

- Negotiable: Mientras no esté incluida dentro de un sprint, puede ser renegociada, cambiada, ampliada las veces que sean necesarias.

- Valuable: Aportar valor al usuario y al negocio

- Estimable: Debe ser comprensible y estimable por los miembros del equipo

- Size or Small: Disponer de un tamaño adecuado para poder priorizarla e incluirla dentro de un sprint.

- Testable: La historia de usuario debe contar con una serie de pruebas o criterios que permitan asegurar que cumplen con su objetivo

2.6.2 Sprint Backlog

El Sprint Backlog son las historias de usuario que se incluyen en el sprint, son los trabajos que el equipo se ha comprometido a realizar en la iteración. El sprint backlog está directamente relacionado por la prioridad asignada por el Product Owner. Aunque la gestión sea realizada directamente por el equipo Scrum, este es el responsable de valorar el coste y ver si realmente encajan dentro del sprint, si son demasiado grandes deben subdividirse.

Cada historia de usuario se divide en tareas, descritas ya en lenguaje más técnico donde se pueden incluir una estimación más detallada. Una vez separadas en tareas y preparadas estarían todas las tareas en estado ñpendienteò, cada vez que un miembro del equipo empieza a trabajar con una, pasa a estado ñen procesoò y una vez que termina pasan a estado ñTerminadoò, los estados se pueden añadir según las necesidades de cada equipo, por ejemplo, se puede disponer de un estado ñpruebasòo un estado ñQAòpara pasar la calidad. Para que todo el equipo pueda visualizar el estado de las tareas se suele disponer en un

tablero, muy parecido a Kanban pero más simplificado, de este modo se tiene una visión más completa del ritmo del proyecto y de las dificultades o cuellos de botella que puedan surgir.

Pendiente	En proceso	Pruebas	QA	Terminado

2.7 Estimaciones

2.7.1 Story Points

Los Story Points o puntos de historia, es la medida que cada equipo Scrum dispone para reflejar el esfuerzo en realizar una historia de usuario, una vez que el punto de historia de define debe mantenerse a lo largo de la duración del equipo.

Lo normal es coger una historia de usuario y definir que se ha tardado en completar por ejemplo, cinco puntos de historia. Esa será la medida a tomar para el resto de historias, se puede traducir más tarde por horas/hombre o días/hombre.

También es importante dejar claro cómo se toman las medidas, no es lo mismo tiempo real incluyendo interrupciones que tiempo ideal donde no se consideran las interrupciones.

2.7.2 Planning Poker

El Planning Póker es la técnica más usada en proyectos Scrum, principalmente por su sencillez, rapidez e implicación de todos los miembros del equipo. El mecanismo es sencillo se necesitan una serie de cartas, principalmente siguiente la secuencia de Fibonacci, para no tener un sinfín de numero de cartas, los miembros del equipo elegirán lo que cada uno considere que se acerca más a lo que se tarda en hacer una tarea. Hay dos cartas diferentes la interrogación (No se tiene el conocimiento o la información para valorarlo) y la taza de café (se necesita un descanso para meditarlo).

El proceso del planning póker discurre de la siguiente manera:

Se presentan las tareas a estimar una a una, se detalla en profundidad en que consiste cada una y se debate por si hay alguna duda. Suele darse un tiempo máximo de discusión

Cada componente individualmente elige su carta lo que considera que puede llevar de trabajo.

Se presentan las cartas de cada uno al mismo tiempo, de esta forma se evita que las estimaciones influyan con las de otra persona. Si las estimaciones son muy diferentes entre sí, unos dicen ½ y otros 8, se vuelve a debatir por si hay algún tipo de malentendido, y se vuelve a estimar.

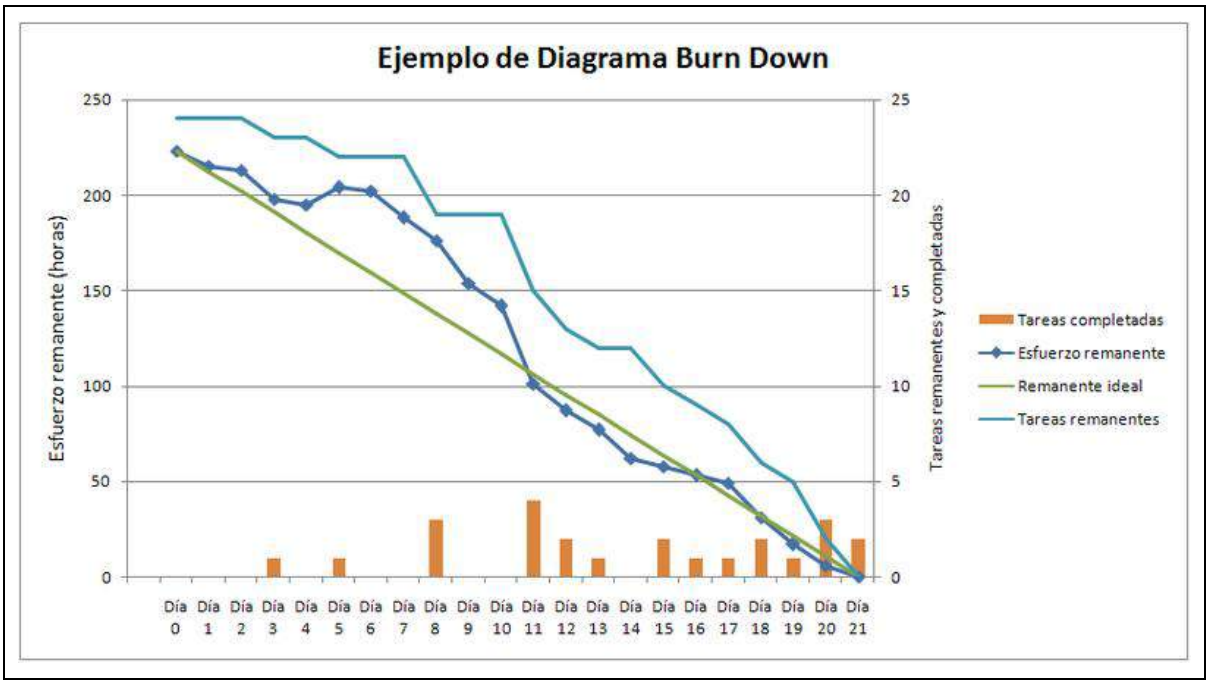
Si no hay una gran diferencia de estimaciones se llega a un consenso y la tarea está estimada.

2.8 Gráfico Burn Down

Es el grafico más utilizado y más útil dentro de los proyectos Scrum, refleja la velocidad del proyecto, comprobando si el proyecto va en los tiempos fijados o está sufriendo retrasos. En el eje de las X figura los días del sprint. En el eje Y, se

refleja el Story Points total del sprint (La suma de todos los puntos de historia). Cada vez que se completa una tarea, se marca un punto justo en el día del sprint que termina (0X), y el punto de Story Points pendientes ï Story Point realizados (0Y). Según se van acabando las tareas el eje Y tiende a cero tareas pendientes, mientras el eje X tiende al último día del Sprint.

Al iniciar la gráfica siempre se dibuja una diagonal recta hacia abajo que representa la línea ideal de trabajo del sprint. Si un punto al acabar una tarea esta debajo de la diagonal significa que se va más rápido del tiempo fijado. Por el contrario, si el punto se encuentra por encima de la línea, hay retrasos y se debe corregir.



2.1 Control de riesgos

El control de riesgos dentro de las metodologías agiles no está definido, pero hay personas que consideran importante tenerlo en cuenta en los proyectos.

A continuación vamos a definir un sistema de registro de riesgos simple y conciso, evitando las técnicas de riesgo tradicionales, más pesadas:

Descripción del riesgo: Descripción breve y clara del riesgo

Fecha de identificación: Fecha de identificación.

Probabilidad: Probabilidad estimada de que ocurra el riesgo.

Severidad: La severidad del riesgo según al impacto.

Prioridad (opcional): Puede ser un valor independiente o un producto de probabilidad y severidad. Una alta severidad con alta probabilidad debe tener más importancia que una de alto riesgo con baja probabilidad.

Propietario: La persona del equipo que controla y toma el control del riesgo

Acción: La respuesta definida para controlar/manejar el riesgo.

Estado: Indica si el riesgo está abierto, cerrado o tratándose.

El riesgo debe ser comunicado en cada sprint meeeting de este modo todo el equipo conoce el riesgo y puede colaborar a controlarlo.

Otra forma de control de riesgos, es con el grafico Burn-down para riesgos. Se toman los siguientes datos:

Riesgo: Breve descripción del riesgo.

Probabilidad: Probabilidad del riesgo

Tamaño de pérdida: Cantidad de tiempo perdido. Puede ser representado en días o puntos de historia.

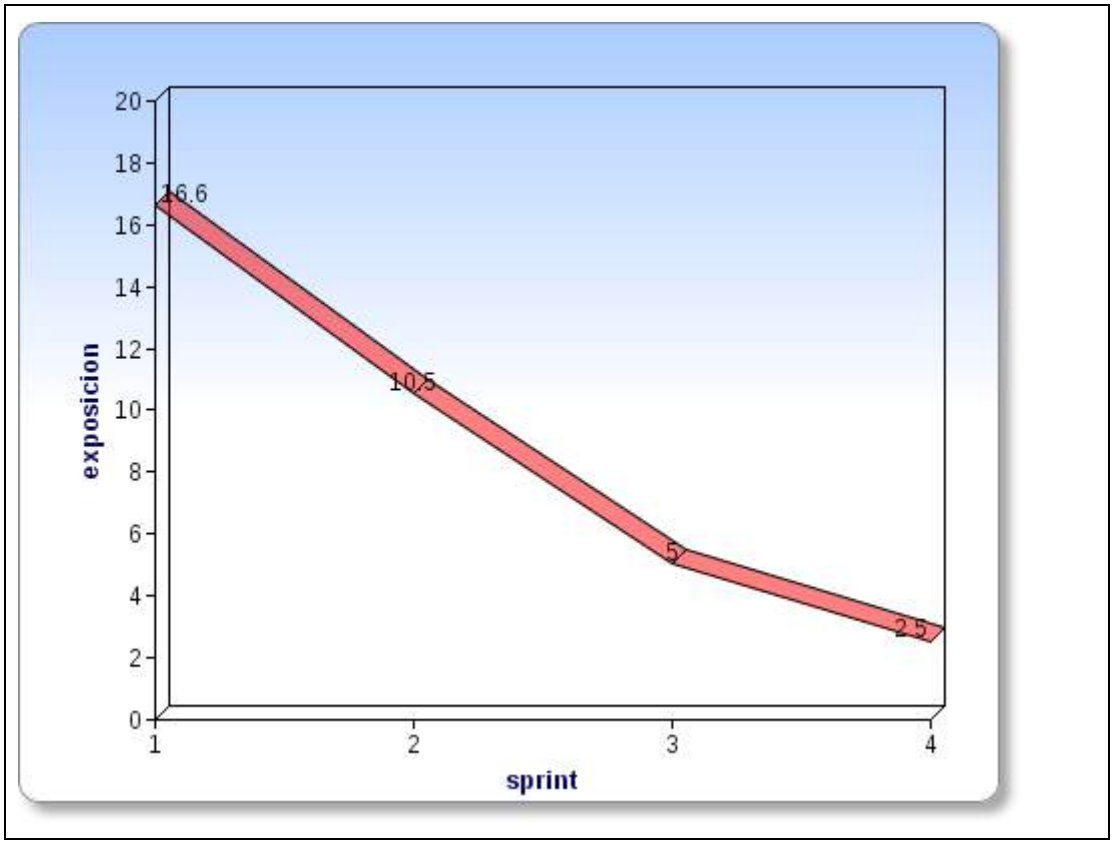
Exposición: El computo entre probabilidad y tamaño de perdida

Riesgo	**Probabilidad**	**Tamaño de perdida**	**Exposición**
Fallo o caída de los	65%	10 días	0,65 * 10 = 6.5

servicios			
Sistemas dependientes no son compatibles durante la integración	80%	12 días	0,8 * 12 = 9,6
El equipo 2, no haya terminado el desarrollo A en el Sprint indicado	5%	10 días	0,05 * 10 = 0,5
			16,6

Una vez evaluados los riesgos de cada Sprint, se van agrupando en una tabla para posteriormente crear el gráfico burn-down, los riesgos cada vez deben ser menores y estar más controlados, con lo cual la gráfica debería tender hacia abajo, esto indica que el proyecto cada vez tiene menos riesgos y mas vigilados.

Sprint	Exposición
1	16,6
2	10
3	5
4	2,5


20
18
16
14
12
10
8
6
4
2
0
exposicion
16.6
10.5
5
2.5
1
2
3
4
sprint

3. Herramientas útiles para un proyecto ágil

3.1 Herramientas de gestión

Hay bastantes herramientas de gestión para proyectos agiles, muchos de ellos online otras de escritorio, se realizara un listado con los más utilizados, también se incluye algunas impresiones iniciales de los que han conseguido probar. En general, la impresión es que son totalmente prescindibles, quitan tiempo y visualización del proyecto, se consigue mayor rapidez con el tablero físico y los posit tradicionales, o haciendo las gráficas burn-down a mano.

sprintometer	http://sprintometer.com/
Aplicación de escritorio libre	Bastante completa, genera varias gráficas. Interfaz parecida a un MS-Project para Scrum o XP, a veces poco intuitivo, cuanta más información se introduzca más detallado.

ScrumDo	http://www.scrumdo.com/
Online gratuita solo tres miembros	Tiene la posibilidad de gráficas, planning póker, dashboard. Lo poco que se ha probado no merece la pena

Kunagi	http://kunagi.org/
Aplicación web, libre.	Tiene la posibilidad de gráficas, dashboard. Basico pero poco intuitivo, se puede montar en un Tomcat o lanzar el ejecutable con un Tomcat embebido

3.2 Herramientas de integración continúa

Las herramientas de integración continua están adquiriendo cada vez más importancia dentro de los proyectos de software, no solamente dentro de proyectos agiles. La integración continua ayuda a tener controlado los cambios realizados a través de pruebas unitarias, incluyen soporte para control de versiones, herramientas de calidad y automatismos que facilitan la detección de errores.

Jenkins	http://jenkins-ci.org/
Aplicación web libre	El proyecto de integración continua, separado del proyecto Hudson de Oracle. El uso de terceras herramientas (control de versiones, pruebas unitarias, herramientas de construccióné) para configurar los proyectos se realiza a través de plugins

Hudson	http://hudson-ci.org/
Aplicación web libre	Prácticamente igual que Jenkins, pero perteneciente a Oracle. El uso de las herramientas que se necesiten se realiza a través de plugins Principales productos que soporta Hudson • Proveedores SCM: Git, CVS, SVN, Perforce, Mercurial, Team Foundation • Herramientas de construcción: Ant, maven, gradle, MSBuild, Nant, Rake • Frameworks de pruebas unitarias: JUnit, NUnit, Selenium, CppUnit, TestNg, XUnit • H. de cobertura de código: Clover, Cobertura, Emma, Serenity, Sonar, NCover, Jacoco • H. de análisis de código: Checkstyle, PMD, Dry, Findbugs, Warnings, CCM, Violations • Herramientas de seguridad: LDAP, Active Directory, Crowd, OpenID

	• Servidores de aplicacion: Weblogic, Glassfish, Tomcat, JBoss, IIS, JRebel • Entornos virtuales: EC2, Virtual Box, VmWare, JCloud • Comunicaciones sociales: E-mail, IRC, Jabber, SMS, Twitter

IBM Rational Jazz	http://www-01.ibm.com/software/rational/jazz/
De pago.	Es la apuesta de IBM por la integración continua, al contrario de las anteriores la plataforma Jazz, tiene sus propias herramientas de control de versiones (Rational Team Concert), de requisitos, pruebasé

4. Conclusiones de la memoria

4.1 Resumen

El contenido del proyecto se ha estructurado en tres bloques: todos relacionados con la gestión de proyectos ágiles y las metodologías más extendidas.

En la primera parte, se realizó una introducción a las metodologías agiles. Fundamentalmente, se han descrito las características principales de las más comunes:

- *Kanban.*
- *Xtreme Programming.*
- *Scrum.*
- *Lean software*.

Se llevó a cabo un estudio sobre puntos más concretos de gestión como: facturación, dificultades reales de implantación y las posibles fusiones entre varias metodologías.

En el segundo bloque del proyecto, se hizo un estudio sobre Scrum y se decidió de este modo porque es, sin lugar a dudas, la más utilizada. Entre los puntos desarrollados se trataron:

- Roles dentro de los equipos Scrum.
- Normas que rigen Scrum.
- Las diferentes reuniones.
- En qué consisten los *Sprin*t.
- Cómo estimar proyectos ágiles.
- Los *backlog*.
- Cómo realizar un gráfico *burn-down*.

Por último, se efectuó un pequeño estudio sobre las herramientas de soporte para la gestión de proyectos ágiles, subdividiendo éste en dos subgrupos:

- Herramientas de gestión.
- Herramientas de integración continua.

Por tanto, el índice del proyecto final se estructura de la siguiente forma:

4.2 Consecución de los objetivos propuestos

En el primer capítulo del proyecto, se ha conseguido desarrollar un análisis bastante aproximado de los objetivos iniciales.

- Se ha elaborado una introducción general del proyecto y de las metodologías ágiles en general.
- La descripción de las diferentes metodologías ha sido más amplia de lo inicialmente esperado, consiguiendo una reflexiva introducción que permite visualizar de una manera general en qué consisten y de qué características propias disponen.
- Por otro lado, se han incluido apartados que, desde el punto de vista del autor, proporcionan valor al proyecto y que, en singulares ocasiones, se pueden consultar en libros o webs especializadas. Estos aspectos se refieren a las posibles facturaciones y las dificultades de implantación de proyectos ágiles.

En el segundo capítulo, el objetivo era centrarse en una descripción al detalle de la metodología Scrum. Se ha decidido ampliar esta metodología porque, sin duda, es la más extendida. En estos objetivos examinados se han incluido:

- Los roles que dispone Scrum. En este sentido, no sólo se han incluido los involucrados directamente en el proyecto, sino además otros perfiles que también tienen interés en los proyectos.
- Se han descrito las diferentes reuniones y puntos de control de obligado cumplimiento, se han detallado en qué momento se deben producir y cómo deben trascurrir. Además, se ha desarrollado el Scrum de Scrum para el control de varios proyectos interconectados.
- Se han detallado todos los artefactos que se incluyen para el control de objetivos y requisitos (backlogs), aparte de añadir diferentes formas para realizar estimaciones ágiles.
- Se han presentado varias gráficas que proporcionan un importante valor visual para el control del proyecto, como es el Burn-out.
- Para finalizar, se han incluido varias formas de control de riesgos en proyectos aplicados a la gestión de proyectos con Scrum. Otro punto

normalmente no tratado en las metodologías ágiles y que da valor añadido al trabajo.

En el último capítulo del proyecto, se confeccionó una investigación de las principales herramientas de ayuda, tanto para la gestión de proyectos ágiles como para el control de calidad y pruebas del software. De esta forma, cualquier lector puede hacerse una idea rápida de las diferentes herramientas que existen y que puede emplear.

4.3 Aspectos formales

Tamaño del papel: vertical A4, 21.0 x 29.7 cm.

Fuente: Arial, Tamaño 12.

Interlineado 1,0 espacio.

Márgenes:

Izquierdo: 3.0 cm.
Derecho: 2.5 cm.
Superior: 4 cm.
Inferior: 3.5 cm.

Títulos:

Titulo capítulo: Arial Narrow 18 negrita, numeración Century Gothic 28. Salto de página en cada capítulo.

Subtítulo de primer nivel: Arial Narrow 15 negrita, numeración Century Gothic 20.

Subtítulo de segundo nivel: Arial 14 negrita, numeración Century Gothic 14.

Párrafos:

Los párrafos deben estar justificados.

Cada capítulo debe comenzar en una nueva página, encabezada con su título y seguida por el texto correspondiente.

El trabajo debe presentarse personalizado con el logo UOC en las cabeceras de cada página en el lado derecho y en los pies de página debe figurar la fecha, el número de página y el total del mismo documento. Así como el número de PEC correspondiente.

La portada de los documentos debe figurar con una separación con la imagen de la mano digital elegida por el autor en la parte superior. En la parte inferior, deberá presentarse el título del documento, la fecha, el autor y el nombre del consultor. En la parte media figura el título del trabajo general.

En la segunda página figuran las dedicatorias del autor y debe presentarse en el margen izquierdo superior con fuente Arial 12 cursiva.

En la tercera página se presenta el índice y la fuente de los títulos de primer nivel es Arial 12 y los de segundo nivel Century Gothic 11.

En el color del índice, los números de los títulos, portada y tablas se ha elegido un color lo más parecido a azul énfasis 1, para dar una sensación de uniformidad y estar acorde con el logo UOC.

Los enlaces de páginas web figurarán con el enlace completo en azul y con hipervínculo, para poder ser visitadas sin dificultades.

4.4 Bibliografía

A continuación, se citan las fuentes consultadas hasta el momento:

- Gestión ágil de proyectos software. Javier Garzás, Juan Enríquez y Emanuel Irrazábal.
- Manual imprescindible de Métodos agiles y Scrum. Alonso Álvarez, Rafael de las Heras y Carmen Lasa.
- Flexibilidad con Scrum. Juan Palacios.
- Scrum Manager: Proyectos. Juan Palacios y Claudia Ruata.
- Scrum y XP desde las trincheras. Henrik Kniberg.
- Wikipedia:

http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software
http://en.wikipedia.org/wiki/Scrum_(development)
http://es.wikipedia.org/wiki/Integraci%C3%B3n_continua
http://es.wikipedia.org/wiki/Kanban
http://es.wikipedia.org/wiki/Programaci%C3%B3n_Extrema

- http://agilemanifesto.org/iso/es/principles.html
- http://www.scrumalliance.org
- http://www.extremeprogramming.org
- http://www.javiergarzas.com/
- http://www.scrum.org

4.5 Contribuciones personales

Las aportaciones personales del proyecto han sido la experiencia en el desarrollo de software: unos diez años en distintos clientes y proyectos (Bankia, Telefónica, BBVA, Consorcio de Transportes de Madrid, El Corte Inglés), trabajando para consultoras como Atos, Iecisa y Everis.

Este bagaje laboral me ha aportado la pericia suficiente para conocer de primera mano la problemática que tiene la implantación de estas metodologías en los proyectos de las grandes compañías. Por un lado, el inconveniente económico ya que rigen los contratos cerrados con fecha establecida y, por otro, la resistencia al cambio, bastante común en empresas jerarquizadas (ñsi se lleva haciendo de este modo toda la vida porque cambiarò).

Asimismo, esta destreza me ha ayudado en el apartado de facturación, porque estoy acostumbrado a la mayoría de contratos materializados que, en España se suelen limitar a ñBodyshoppingòy proyectos ñLlave en manoò

El resto de apartados expuestos han germinado gracias a la investigación, indagación y los compañeros certificados en Scrum.

Como conclusión, puedo aseverar que este estudio me ha permitido investigar con hondura en qué consisten las metodologías ágiles y su gestión. Por tanto, considero que el aprendizaje ha sido instructivo, eficaz y de utilidad.